Rebalancing Collection Indexes Approach

Goal

Move granules in large collections out of the small collections index into their own index to improve search performance

Traceability

⚠ CMR-2543 - JIRA project doesn't exist or you don't have permission to view it.

This is the approach we'll take to resolve unbalanced collection indexes and issues described in ⚠ CMR-2537 - JIRA project doesn't exist or you don't have permission to view it. .

Requirements/Criteria

- Move larger collections to their own index.
- Search results should be correct during and after a move.
- No downtime.
- Don't run out of disk space or memory within Elasticsearch
- Should be straightforward and safe for an CMR Operations to run.
- Must document the process for CMR Operations

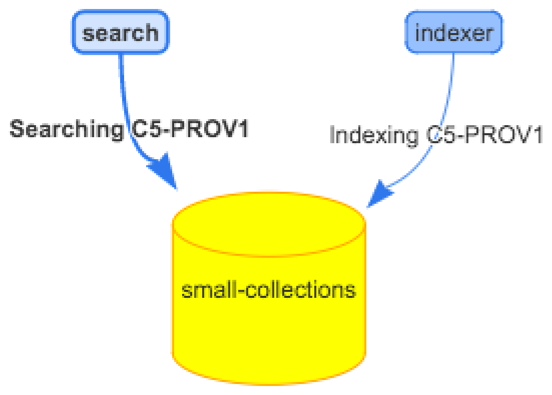
Process

1. Kickoff move collection using bootstrap
2. Monitor bootstrap logs/wait for email.
3. Finalize move using bootstrap
4. Verify move

The steps are described below showing the process for moving an example collection, C5-PROV1, into it's own index.

Before

The initial state is C5-PROV1 granules are in the small collections index.

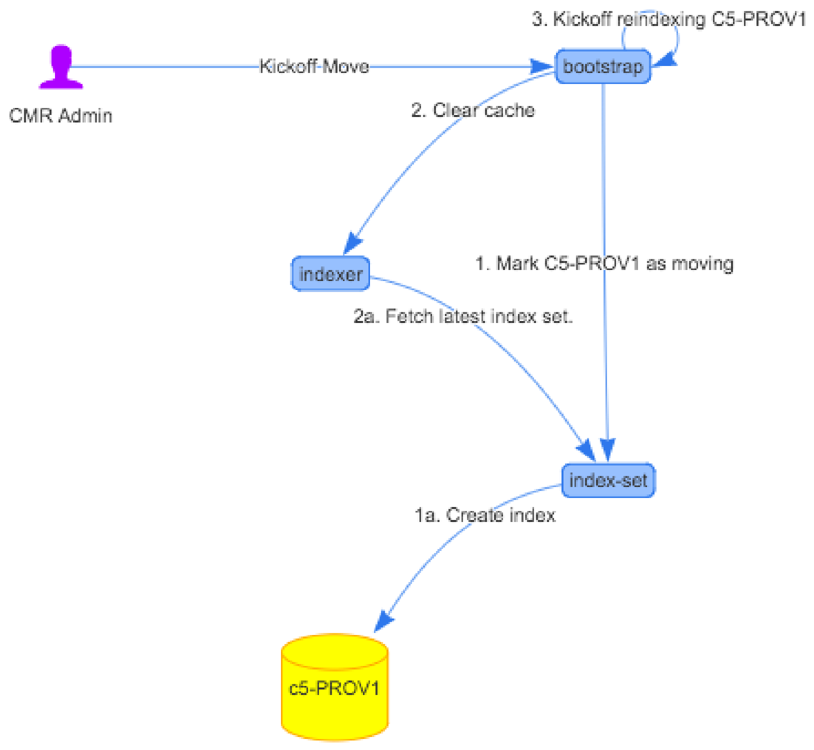


1. Kickoff move collection using bootstrap

Bootstrap will have a new endpoint for orchestrating the moving of a collection to it's own index. It will do the following.

1. Call index set endpoint to add moving collection. (new endpoint)
    a. This will mark the collection as moving in the index set data and create the collection index.
2. Clear indexer cache (just for index set related caches. Not supported yet. Could also work with all caches.)
    a. We'll update indexer to use a consistent cache so that all instances will be cleared with one request.

3. Kick off reindexing of the collection within the current bootstrap.

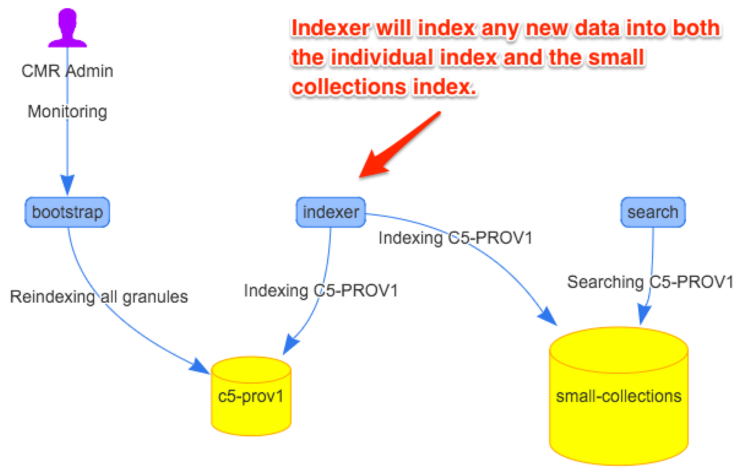


2. Monitor Progress

At this point the services should be doing the following:

- Search - The search application isn't impacted by the collection being moved. It should continue to search the small collections data.
- Indexer - Granules updated on this collection will be written to both the small collections index and the separate index.
- Bootstrap - Reindexing granules in C5-PROV1 into the new index.

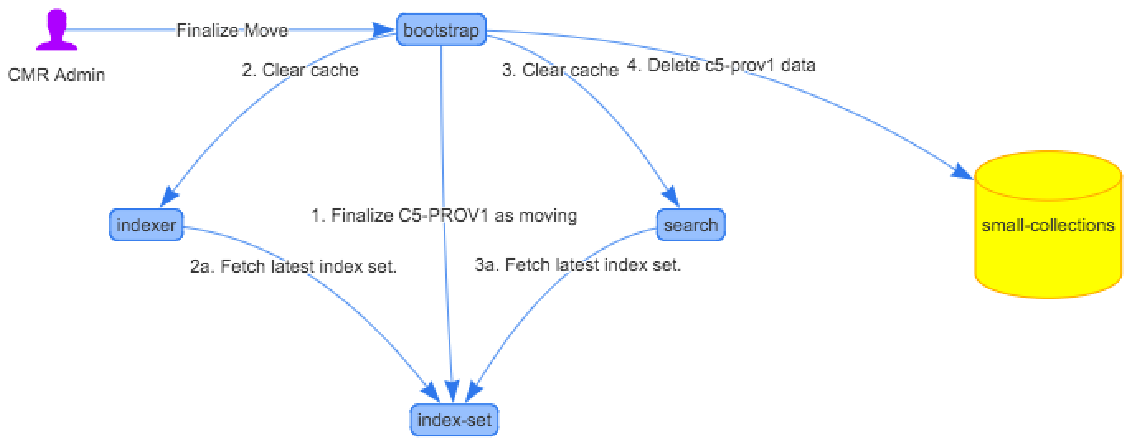
We expect that multiple collection moves can be done at the same time on separate bootstrap instances. We will need to test this.



3. Finalize move using Bootstrap

Bootstrap will finalize the move at the end to remove the old data and switch search and other apps to use the new index only.

1. Call index set endpoint to finalize moving collection. (new)
* This will unmark the collection as moving and add it to the list of collections with their own index.
2. Clear indexer cache (just for related index set related cache)
3. Clear search cache (just for related index set related cache) UPDATE: We will no longer do this step. We've changed search to use a background job to keep this up to date.
4. Delete the data from small collections.

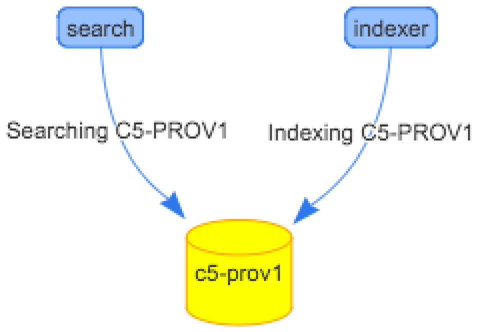


Finalize Move Race Condition

There's a race condition during finalize. After the index set has been updated in step 1 and the indexer cache has been cleared in step 2 there's a period of time during which the different indexer applications may be processing granules for this very collection and may have already decided which index its going to. It's possible that the indexer will index a granule into small collections after the bootstrap has issued the delete. The next step to verify should identify if the race conditions has occurred. We will change the caching of index set information to use a consistent cache so that clearing that cache on any instance will cause all instances to clear. We could mitigate the chance of the race condition occurring by adding a short sleep that would allow any running indexers to complete. A sleep of 5 seconds should be enough for most cases. Note that this just helps reduce the chance of the race condition. but does not avoid the race condition.

After Finalize

At this point both search and indexer will be searching and writing to the new collection index. The small collections index will not contain any granules for the collection that was moved (unless in the case of the race condition which will be checked during verification).

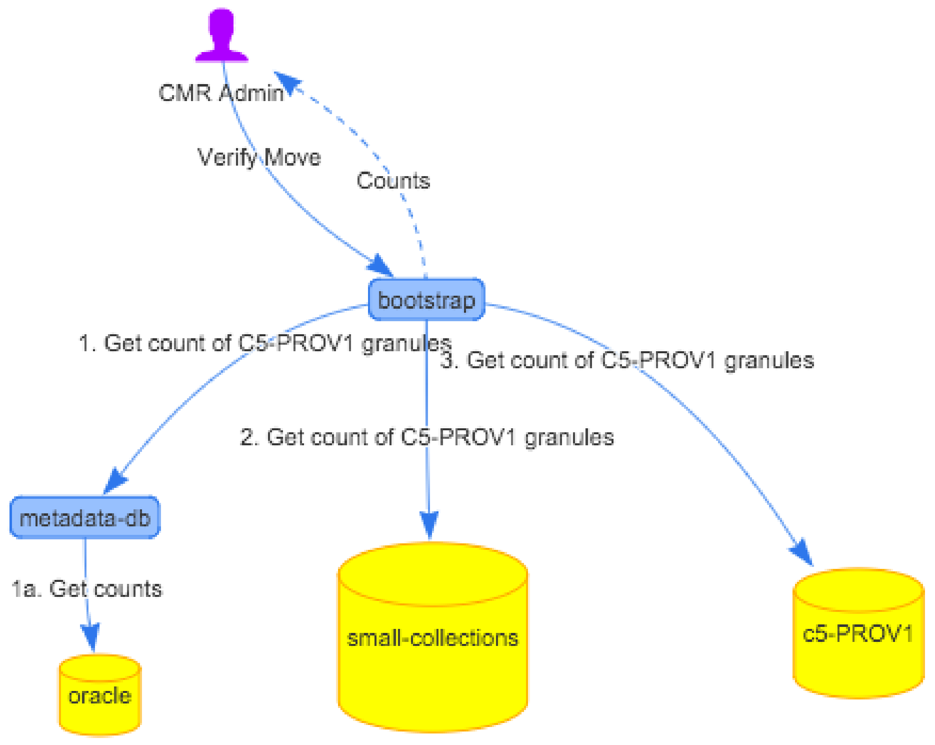


4. Verify Move

We need to make sure that the move was completed successfully from small collections into the new index. Verify will get the following information and return it.

* Count of granules in the collection in the database.
* Count of granules in the collection in the small collections index.
* Count of granules in the collection in the separate index.

We expect the # of granules in the db and separate index to be close to the same if there is ongoing ingest or identical if there's no recent ingest. The # of granules in small collections should be 0 for this collection. If we find that it's not 0 we'll need to investigate. The resolution may be to manually perform the delete from small collections.


CMR Admin
Verify Move
Counts
bootstrap
1. Get count of C5-PROV1 granules
2. Get count of C5-PROV1 granules
3. Get count of C5-PROV1 granules
metadata-db
1a. Get counts
oracle
small-collections
c5-PROV1

CMR Changes

Splunk Changes

- Add alert for when bootstrap reindex of a collection has completed.

Common App Changes

- Add ability to clear a specific cache on the cache api.

Index Set Changes

- Mark Collection as Moving Operation
    - This operation would update the index set data to mark a collection as being moved out of small collections. It would update the stored index set data and create the collection index.
- Finalize Collection Move Operation
    - This operation would update the index set to take a moving collection and update the index set so that it is no longer marked as moving and represented as a normal collection index.

Bootstrap Changes

- Kickoff Move Operation
    - Performs actions described above.
- Finalize Move Operation

    - Performs actions described above.
- Ability to reindex a collection into only the collection index.
    - The collection will be in the index set both as in small collections and in its own index. Bootstrap will need to be able to reindex granules into just the new index. This will most likely be a combination change in the indexer and bootstrap.
- Verify Operation
    - Retrieves counts as described above

Search Changes

- We originally going to use a consistent cache for search but that would mean every search request has to make a request to cubby. That would be very fast but we decided to reduce that. Search isn't required to be absolutely consistent here. We set up search with a job to refresh that part of the cache every 5 minutes. That means within 5 minutes of the index set changing search should be using the lates information.

Indexer Changes

- Change cached index set data to use the consistent cache. We want all instances to update this if one of them detects it is different or the cache is cleared.
- Update indexing to handle collections marked as moving to their own index.
    - It must index granules in those collections both into small collections and into their own collection.
- Remove configuration for collections with separate indexes.
    - This will be changed to just use whatever is in the index set. Collections will only be split into their own indexes through the process described on this page.

Integration Testing

We will integration test these changes with an automated test locally. This is the test plan.

1. Create several sample collections and granules in each collection.
2. Verify counts for all collections are in small collections.
    a. When verifying counts we should use the verify endpoint as well as retrieving provider holdings. This will check that the search application sees the right indexes and doesn't report duplicates.
3. Kickoff move operation for a collection in synchronous mode
    a. A response will be returned once the collection's data has been reindexed.
4. Verify counts of the collection. The granule counts should be the same for all indexes.
5. Ingest new data. (The data should be indexed into both indexes)
6. Verify counts of the collection. The granule counts should be the same for all indexes and increased corresponding to the number of granules indexed.
7. Finalize the collection move
8. Verify counts of the collection. The granule counts for the moved collection in small collections should be 0.
9. Ingest new data. (The data should be indexed only into the new index)
10. Verify counts of the collection. The granule counts for the moved collection in small collections should be 0.

Rollout Plan

1. Review this design
2. Create stories/tasks for the work and put into a sprint.
3. Implement all of the stories including the integration testing above.
4. Test in workload
    a. The workload test will be similar to the integration testing except that we will run ingest concurrent through some of the stages.
5. Test in SIT
    a. I'm not sure if we'll test here given we don't have any collection specific indexes. There aren't that many granules in SIT.
6. Deploy and Test in UAT
    a. CMR Ops will verify they can perform the steps in UAT.
7. Deploy to Ops
8. Perform move of a single collection in operations
    a. We'll closely monitor the first time to make sure it goes ok.
9. Move all large collections into their own indexes in operations.

Collections in Operations To Separate

Current Size Metrics

Number of shards in database now: 1472

- Marvel: 28
- Tags: 10
- Cubby: 6
- Groups: 6
- Index Sets: 2
- Collections: 10
- All collection revisions: 10
- Small collections (granules): 40
- Separate collection indexes (granules: 136 indexes * 5 shards * 2 replicas = 1360 shards

Number of granules in small collections: 92,311,605

Number of granules per shard: 4,615,580

After Moving out Collections

Number of granules to move out (based on table below): 47,304,339

New Size: 45,007,266

Number of granules per shard in small collections: 2,250,363

Number of new shards: (num collections in table * 5) => 43 * 5 *2 => 430

Total Number of shards: 1472 + 430 = 2002

These are all the collections in small collections with more than 500,000 granules.

concept id	granule count
C1000000320-LPDAAC_ECS	2782162
C1200034421-OB_DAAC	2255134
C203669731-LPDAAC_ECS	1860220
C203669732-LPDAAC_ECS	1859882
C193529902-LPDAAC_ECS	1850983
C193529903-LPDAAC_ECS	1691677
C193529945-LPDAAC_ECS	1655592
C203669713-LPDAAC_ECS	1601751
C203669714-LPDAAC_ECS	1601701
C193529459-LPDAAC_ECS	1595443
C193529460-LPDAAC_ECS	1457158
C193529462-LPDAAC_ECS	1433017
C203669662-LPDAAC_ECS	1288664
C203669716-LPDAAC_ECS	1288660
C1000001166-NSIDC_ECS	1171567
C1000001182-NSIDC_ECS	1169774
C1000000541-NSIDC_ECS	1152150
C203669700-LPDAAC_ECS	1030677
C203669661-LPDAAC_ECS	1030677
C1000001160-NSIDC_ECS	1010206

C1000001202-NSIDC_ECS	1009887
C1000001168-NSIDC_ECS	1009271
C194001236-LPDAAC_ECS	995342
C188030686-GSFCS4PA	905593
C1206485320-ASF	891952
C1206485940-ASF	891452
C1000001203-NSIDC_ECS	874536
C1200034420-OB_DAAC	824102
C1200034400-OB_DAAC	824102
C194001216-LPDAAC_ECS	798132
C16893873-LPDAAC_ECS	746488
C185174181-USGS_EROS	685064
C190465571-GSFCS4PA	661268
C184697082-GSFCS4PA	661064
C1206936391-ASF	652589
C198883106-GSFCS4PA	515998
C198883271-GSFCS4PA	515998
C28466910-LPDAAC_ECS	515932
C28466913-LPDAAC_ECS	515582
C28466907-LPDAAC_ECS	507405
C1206156901-ASF	505328
C1206487217-ASF	505328
C1206487504-ASF	504831

Error rendering macro 'pageapproval' : null